

大阪フルリージョン 2 周年！  
最新情報からメインリージョンとしての活用事例まで教えます！

# 一步踏み込んで考える Resilience

アマゾン ウェブ サービス ジャパン合同会社

ソリューションアーキテクト

多田 慎也



# 自己紹介

名前：多田 慎也

所属：アマゾンウェブサービスジャパン合同会社  
ソリューションアーキテクト

経歴：インフラエンジニア@Sler

-> 西日本担当のソリューションアーキテクト@AWS

好きなAWSサービス：AWS Step Functions, AWS Fargate



# 本セッションの想定聴講者とゴール

## 想定聴講者

- AWS の基本的なサービス、概念を知っている方
- システムの信頼性 / 可用性設計に興味がある方
- 東京リージョンと大阪リージョンを使ったマルチリージョン構成に興味がある方

## ゴール

- Resilience（回復力）の責任共有モデルを理解できるようになる
- AWS 上で Resilience（回復力）を備えたシステム的设计ポイントが理解できるようになる

# アジェンダ

- なぜ Resilience が重要なのか？
- 責任共有モデル for Resilience
- Resilience のあるシステムを作るためのベストプラクティス
- マルチリージョン基本
- まとめ

# アジェンダ

- なぜ Resilience が重要なのか？
- 責任共有モデル for Resilience
- Resilience のあるシステムを作るためのベストプラクティス
- マルチリージョン基本
- まとめ

# Resilienceとは

英語で、「回復力」、「抵抗力」、「弾力性」を意味する言葉。

AWS における Resilienceとは

ワークロードが障害に対応し、障害から迅速に復旧する能力

A black and white portrait of Werner Vogels, CTO of Amazon.com. He is a middle-aged man with a beard and mustache, wearing a dark jacket over a dark t-shirt. He is looking slightly to the right of the camera with a slight smile. The background is a blurred outdoor setting with trees and a path.

**“Everything fails, all the time.”**

**Werner Vogels**  
CTO, Amazon.com



*ed the Turing tes*

# なぜ Resilience が重要なのか？

## ビジネスへの影響

- 業務遂行に対する直接的な影響  
メール・チャット、生産システム、ATM、etc
- 企業や組織ブランドへの影響

## 課題

- これまで以上に Resilience に沿った設計指針が求められている



# アジェンダ

- なぜ Resilience が重要なのか？
- 責任共有モデル for Resilience
- Resilience のあるシステムを作るためのベストプラクティス
- マルチリージョン基本
- まとめ

# 責任共有モデル for Resilience

お客様

クラウド内のレジリエンスに対する責任

Resilience 'in'  
the cloud

AWS

クラウドのレジリエンスに対する責任

Resilience 'of' the  
cloud

重要なインフラストラクチャの継続的なテスト

ワークロードアーキテクチャー

変更管理 / 運用

Observability / 障害管理

ネットワーク / クォータ / 制限管理

ハードウェア / AWS サービス

コンピュート

ストレージ

データベース

ネットワーキング

AWSグローバルインフラストラクチャー

リージョン

アベイラビリティゾーン

エッジロケーション

# 責任共有モデル for Resilience

お客様

クラウド内のレジリエンスに対する責任

重要なインフラストラクチャの継続的なテスト

ワークロードアーキテクチャー

変更管理 / 運用

Observability / 障害管理

利用者は、**Resilience of the Cloud**の性質を理解し、  
障害への影響をコントロールするため、  
適切な**Resilience in the Cloud**を実践することが重要

クラウドのレジリエンスに対する責任

**Resilience 'of' the cloud**

AWSグローバルインフラストラクチャー

リージョン

アベイラビリティゾーン

エッジロケーション

# Resilience of the Cloud



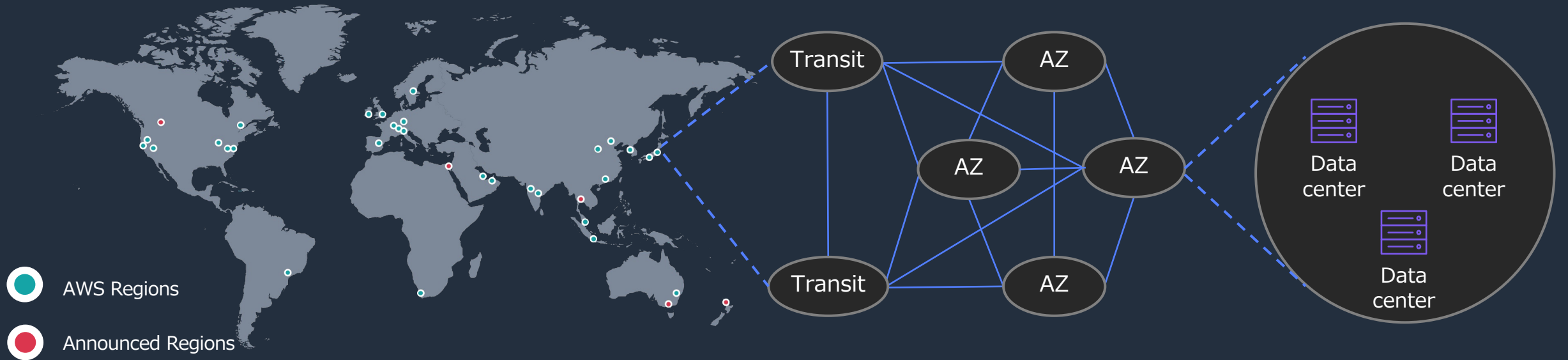
# AWS Regions and Availability Zones (AZs)

AWS REGIONS ARE PHYSICAL LOCATIONS AROUND THE WORLD WHERE WE CLUSTER DATA CENTERS

全世界 31 のリージョン

それぞれのリージョンは、複数アベイラビリティゾーン (AZ) で構成されている

各 AZ は 1 つ以上のデータセンターで構成される



AWS のリージョンは、  
他リージョンと完全に分離されるように設計  
リソースは指定したリージョンに  
結びつけられている

相互に関連する障害を防ぐため、  
最大 60 マイル (約 100 km) 離れており、  
商用電源、断水、ファイバー分離、地震、火災  
、竜巻、洪水などを同時に受けない設計

データセンターは、  
それぞれ冗長化した、ネットワーク  
、接続、および電源を備えている

# Resilience in the Cloud



# 適切に Resilience in the Cloud を設計するには？

## High Availability

設計および運用メカニズムによる一般的な障害に対する耐性



コア サービス、可用性の目標を達成するための設計

## Disaster recovery

まれではあるが影響の大きい障害について、目標内に運用に戻る



バックアップ、データ管理、RTO/RPOの管理

## Resilience の継続性

CI/CD、Code の改善、運用テスト、Observability / モニタリング、カオスエンジニアリング

# 可用性について検討する

可用性とは？：ワークロードが使用可能な時間の割合

アプリケーションの可用性における一般的な設計目標 例：

可用性	最大利用不可能時間 (年間)	アプリケーションのカテゴリ
<u>99%</u>	3 日と 15 時間	バッチ処理、データの抽出、転送、ロードのジョブ
<u>99.9%</u>	8 時間 45 分	ナレッジ管理、プロジェクト追跡などの社内ツール
<u>99.95%</u>	4 時間 22 分	オンラインコマース、POS
<u>99.99%</u>	52 分間	動画配信、ブロードキャストのワークロード
<u>99.999%</u>	5 分間	ATM トランザクション、通信のワークロード

[https://docs.aws.amazon.com/ja\\_jp/wellarchitected/latest/reliability-pillar/availability.html](https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/reliability-pillar/availability.html)





# 可用性を向上させるには？

$$\text{可用性} = \frac{\text{使用可能な時間}}{\text{全体の時間}} = \frac{\text{MTBF (平均故障間隔)}}{\text{MTBF (平均故障間隔)} + \text{MTTR (平均復旧時間)}}$$

MTBF(平均故障間隔)とMTTR(平均復旧時間)を向上・改善するアプローチが必要

# MTBF / MTTR 向上のアプローチ

MTBF (平均故障間隔)を長くする：**耐障害性向上のアプローチ**

- インスタンス/コンポーネントの冗長化(マルチ AZ の活用)
- 過負荷の防止(キャパシティと使用率の確認・負荷制限)
- テストを通じてソフトウェアのバグや欠陥を排除
- 障害範囲の限定化

MTTR (平均復旧時間)を短くする：**障害復旧、運用性向上のアプローチ**

- モニタリング (影響範囲と運用状態の指標を監視)
- 再起動時間が短いサービスの検討 (コンテナ・サーバレス等)
- 復旧手順の確立、復旧対応の自動化

# 適切に Resilience in the Cloud を設計するには？

## High Availability

設計および運用メカニズムによる一般的な障害に対する耐性



コア サービス、可用性の目標を達成するための設計

## Disaster recovery

まれではあるが影響の大きい障害について、目標内に運用に戻る



バックアップ、データ管理、RTO/RPOの管理

## Resilience の継続性

CI/CD、Code の改善、運用テスト、Observability / モニタリング、カオスエンジニアリング

# Disaster Recovery (DR) 目標について検討する

- RTO (目標復旧時間) : サービスの中断からサービスの復旧までの最大許容遅延時間
- RPO (目標復旧時点) : 最後のデータ復旧ポイントからの最大許容時間

どれだけのデータを再作成または、損失する余裕があるか？

Recovery Point objective (RPO)

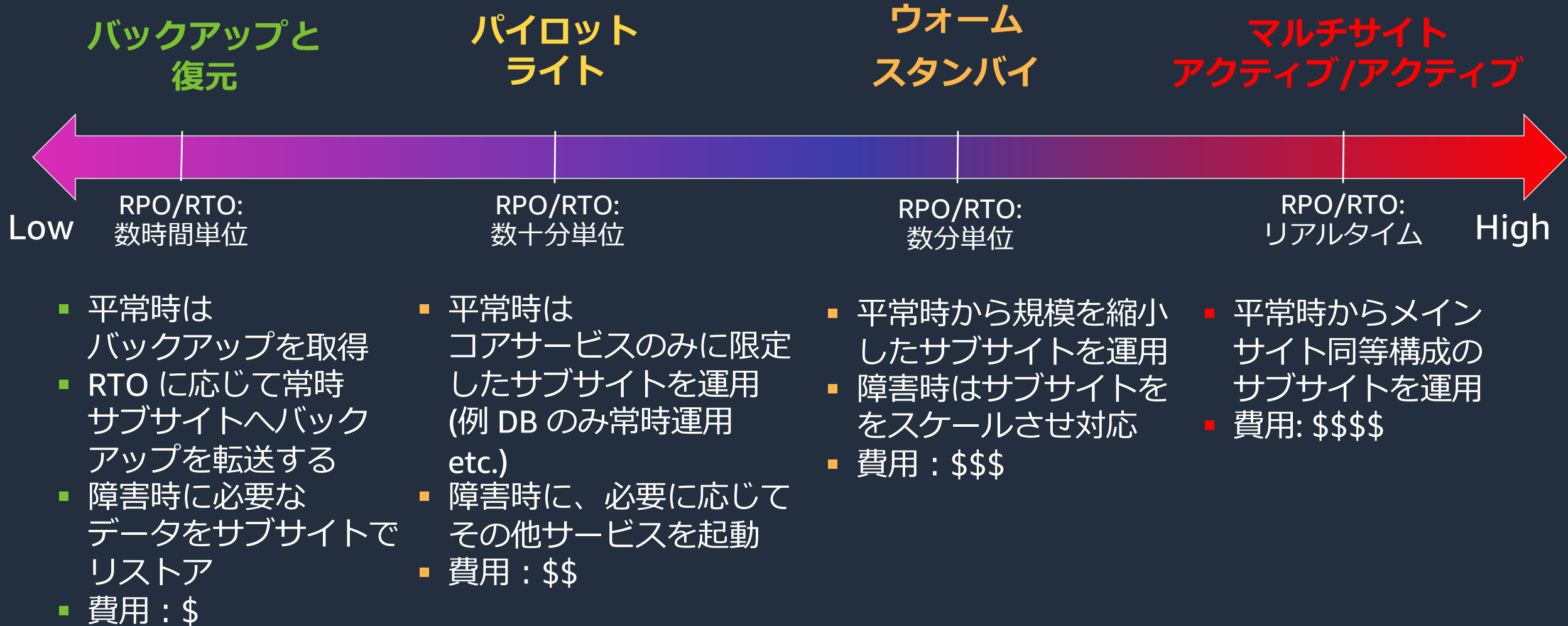
災害

どれくらいのスピードで復旧しなければならないか？  
ダウンタイムにかかるコストは？

Recovery Time objective (RTO)



# AWS におけるディザスタリカバリ (DR) 戦略



# 可用性 / DR のニーズ

## コンポーネント単位でも検討する

宅配ドライバーの  
配送アプリの例



システムの重要性

高

中

低

可用性 / DR

マルチ AZ / パイロットライト

マルチ AZ / バックアップ

マルチ AZ / シングルリージョン



システム全体に対して、均一の目標設定をするのではなく、それぞれのコンポーネントにて、設計目標を**差別化**する

# アジェンダ

- なぜ Resilience が重要なのか？
- 責任共有モデル for Resilience
- Resilience のあるシステムを作るためのベストプラクティス
- マルチリージョン基本
- まとめ

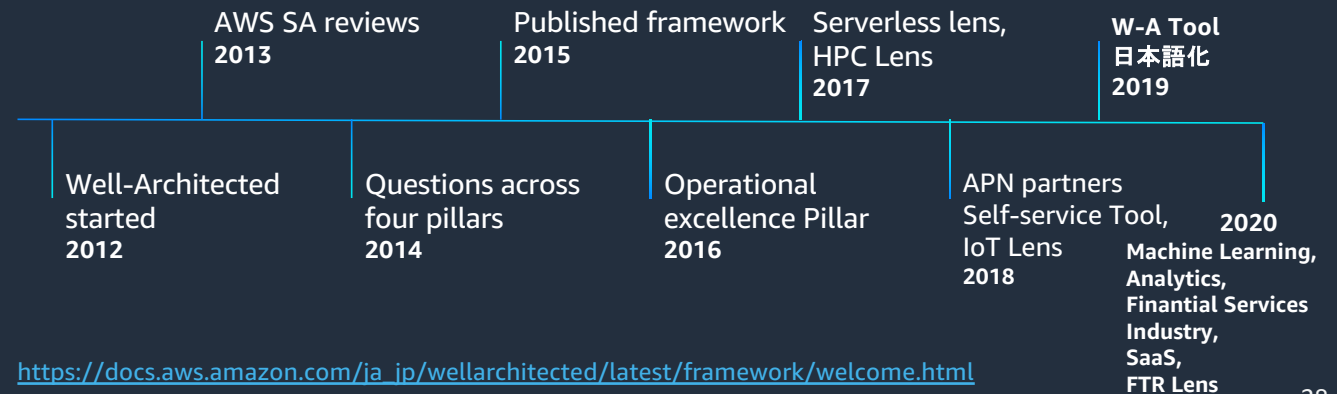
# AWS Well-Architected フレームワーク (W-A) とは?

## システム設計・運用の”大局的な”考え方と ベストプラクティス集

- AWS のソリューションアーキテクト (SA)、パートナー様、お客様の 10 年以上にわたる

**経験**から作り上げたもの

- AWS と**お客様**と共に、  
W-A も**常に進化し続ける**



[https://docs.aws.amazon.com/ja\\_jp/wellarchitected/latest/framework/welcome.html](https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/framework/welcome.html)





# AWS Well-Architected フレームワークの 6 本の柱

## AWS Well-Architected フレームワーク全体の設計原則



運用上の  
優秀性



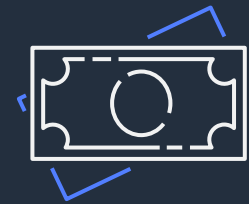
セキュリティ



信頼性



パフォーマンス  
効率



コスト  
最適化



持続可能性

# 信頼性における質問

分野	質問のトピック
基盤	REL 1 サービスクォータと制約はどのように管理しますか?
	REL 2 ネットワークトポロジをどのように計画しますか?
ワークロードアーキテクチャ	REL 3 ワークロードサービスアーキテクチャをどのように設計すればよいですか?
	REL 4 障害を防ぐために、分散システムでの操作をどのように設計すればよいですか?
	REL 5 障害を緩和または耐えるために、分散システムの操作をどのように設計しますか?
変更管理	REL 6 ワークロードリソースをモニタリングするにはどうすればよいですか?
	REL 7 需要の変化に適応するようにワークロードを設計するには、どうすればよいですか?
	REL 8 変更はどのように実装するのですか?
障害管理	REL 9 データはどのようにバックアップするのですか?
	REL 10 ワークロードを保護するために、障害分離をどのように使用すればよいですか?
	REL 11 コンポーネントの障害に耐えるようにワークロードを設計するにはどうすればよいですか?
	REL 12 信頼性はどのようにテストすればよいですか?
	REL 13 災害対策 (DR) はどのように計画するのですか?

# 信頼性における質問

分野	質問のトピック
基盤	REL 1 サービスクォータと制約はどのように管理しますか?
	REL 2 ネットワークトポロジをどのように計画しますか?
ワークロードアーキテクチャ	REL 3 ワークロードサービスアーキテクチャをどのように設計すればよいですか?
	REL 4 障害を防ぐために、分散システムでの操作をどのように設計すればよいですか?
	REL 5 障害を緩和または耐えるために、分散システムの操作をどのように設計しますか?
変更管理	REL 6 ワークロードリソースをモニタリングするにはどうすればよいですか?
	REL 7 需要の変化に適応するようにワークロードを設計するには、どうすればよいですか?
	REL 8 変更はどのように実装するのですか?
障害管理	REL 9 データはどのようにバックアップするのですか?
	REL 10 ワークロードを保護するために、障害分離をどのように使用すればよいですか?
	REL 11 コンポーネントの障害に耐えるようにワークロードを設計するにはどうすればよいですか?
	REL 12 信頼性はどのようにテストすればよいですか?
	REL 13 災害対策 (DR) はどのように計画するのですか?

# REL11. コンポーネントの障害に耐えられるようにワークロードを設計する

## 説明

高い可用性と低い平均復旧時間 (MTTR) の要件を持つワークロードは、回復力を考慮した設計をする必要があります

## ベストプラクティス

REL11-BP01 ワークロードのすべてのコンポーネントをモニタリングして障害を検知する

REL11-BP02 正常なリソースにフェイルオーバーする

REL11-BP03 すべてのレイヤーの修復を自動化する

REL11-BP04 復旧中はコントロールプレーンではなくデータプレーンを利用する

REL11-BP05 静的安定性を使用してバイモーダル動作を防止する

REL11-BP06 イベントが可用性に影響する場合に通知を送信する

# ベストプラクティスの捉え方

ベストプラクティスにすべて沿っている  
必要はありません  
重要なのは、ベストプラクティスを理解し  
リスクや改善点を認識することです。

# アジェンダ

- なぜ Resilience が重要なのか？
- 責任共有モデル for Resilience
- Resilience のあるシステムを作るためのベストプラクティス
- マルチリージョン基本
- まとめ

“マルチリージョンアーキテクチャで  
作る必要があります。”



# まずはリージョン内の Resilience 向上

- マルチリージョンの前に、リージョン内の Resilience 向上から始めましょう
  - リージョン内で Resilience を高めるエンジニアリングを行っていますか？
  - リージョン内で適切なオペレーションを行っていますか？



# マルチリージョン 基本

- 要件を理解する
- データを理解する
- 依存性を理解する
- オペレーションレディネス

# マルチリージョン 基本

- 要件を理解する
- データを理解する
- 依存性を理解する
- オペレーションレディネス

# ビジネス要件の理解

- 極めて高い回復力の要件が求められますか？
  - 目標を満たさない場合の影響は？
- ワークロードが別のリージョンで実行できることを証明できる必要がありますか？
  - 規制ガイド的アプローチ？
- 要件とコストと複雑さを比較検討する

Tier	Max RTO	Max RPO	Criteria	Cost
Platinum	15 min	5 min	ミッションクリティカル	\$\$\$
Gold	15 min – 6 hrs	2 hrs	重要だがミッションクリティカルではない	\$\$
Silver	6 hrs – few days	24 hrs	ミッションクリティカルでない	\$

# 正しい理由でマルチリージョンを実行していますか？



可用性の向上



規制上の要求

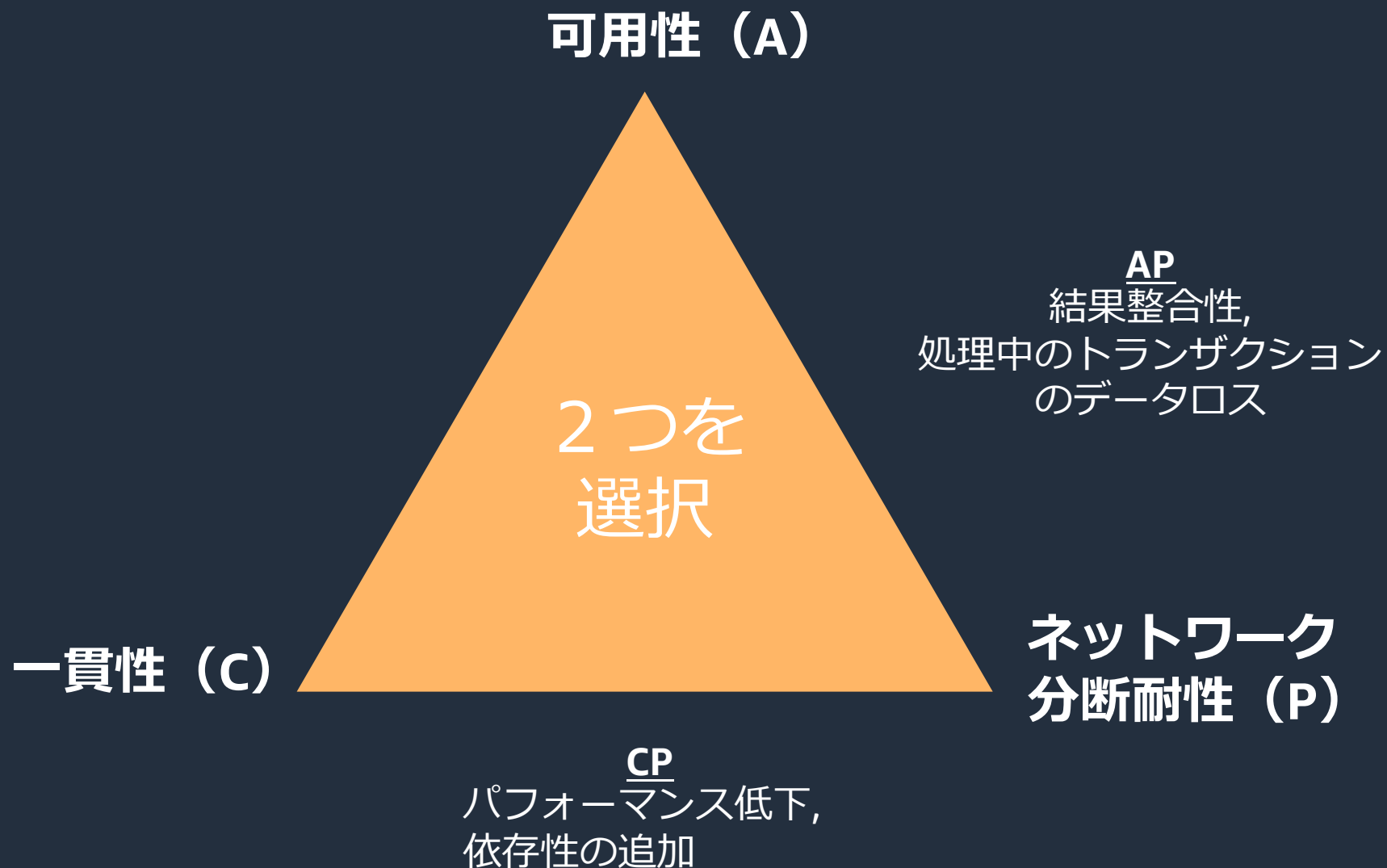


性能の改善

# マルチリージョン 基本

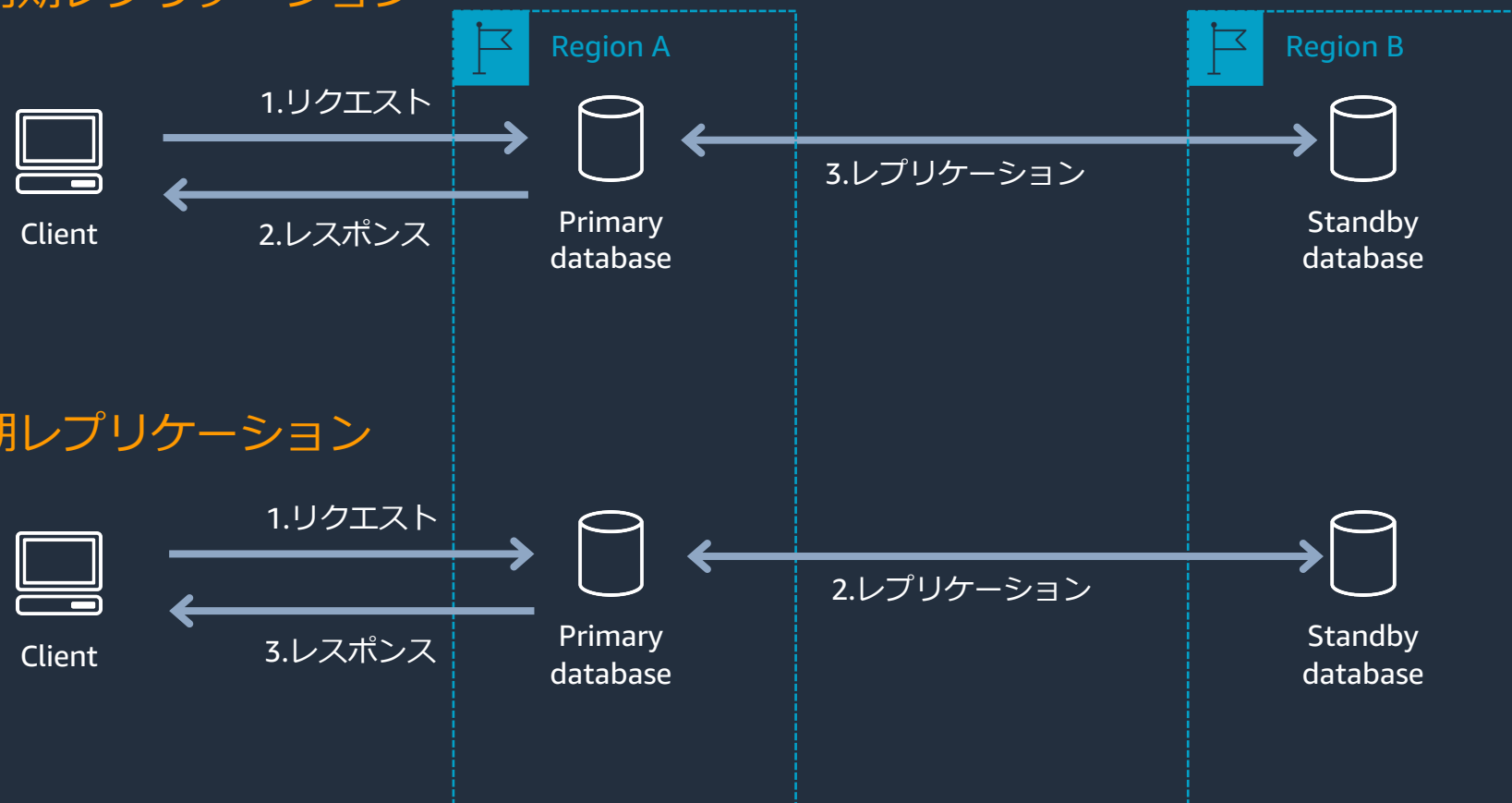
- 要件を理解する
- データを理解する
- 依存性を理解する
- オペレーションレディネス

# CAP定理

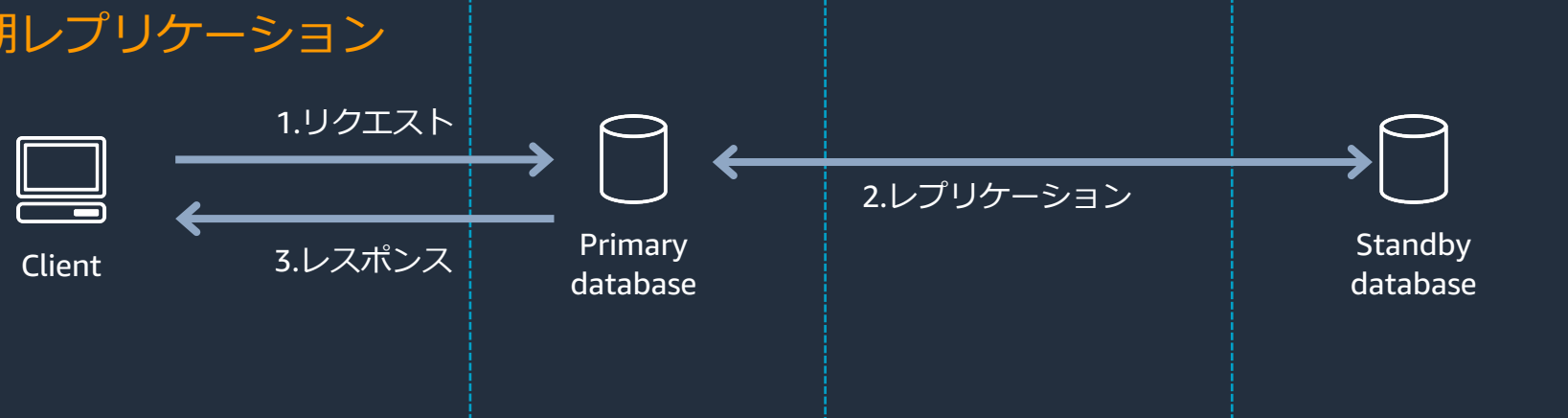


# データの一貫性の要件

## 非同期レプリケーション

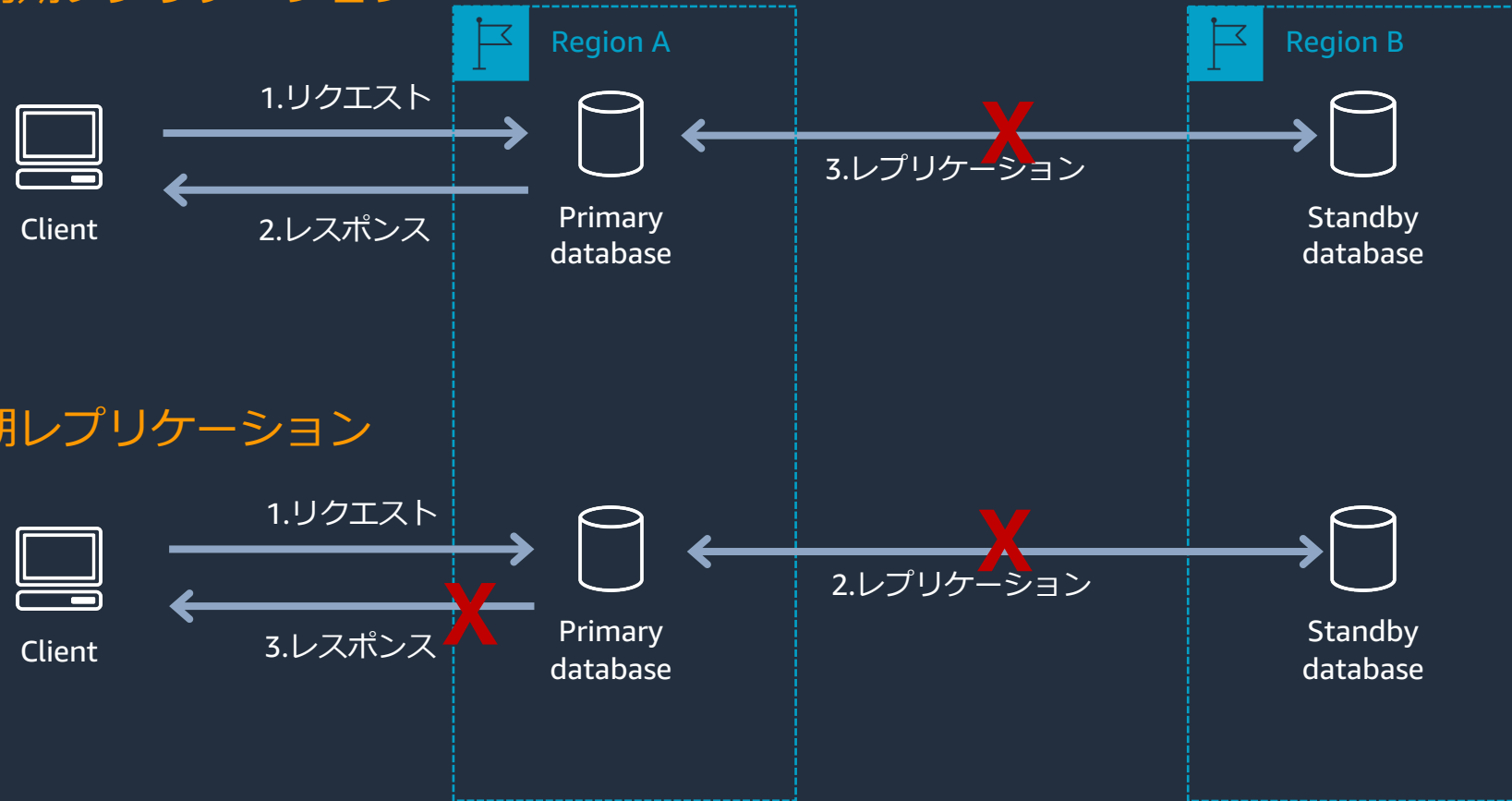


## 同期レプリケーション



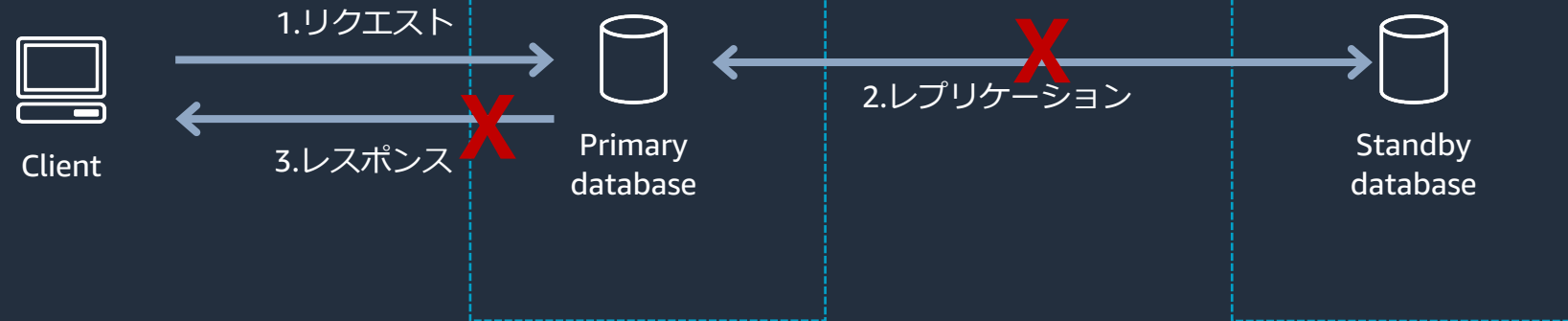
# データの一貫性の要件（NW分断障害）

## 非同期レプリケーション



可用性を優先

## 同期レプリケーション



一貫性を優先



# データのアクセスパターン

## アクセス特性を理解する

### Read Heavy



Client



Database

### Write Heavy



Client



Database

### MIX



Client



Database

Read  
Write

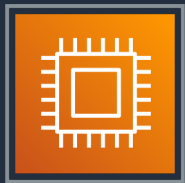


# マルチリージョン 基本

- 要件を理解する
- データを理解する
- 依存性を理解する
- オペレーションレディネス

# AWS サービス

- 使用するリージョンで利用できるサービスを決定する
- マルチリージョンに役立つ機能は何か特定する



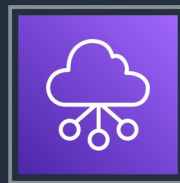
Compute



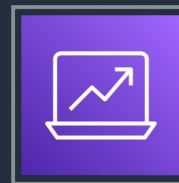
Storage



Database



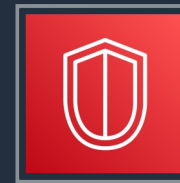
Network  
and  
Content Delivery



Analytics



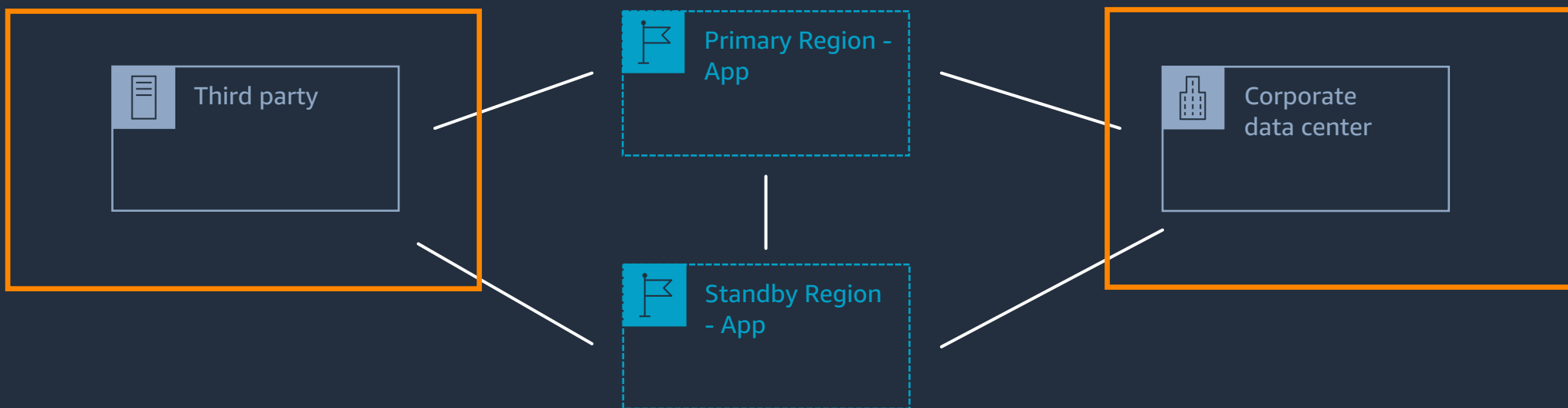
Management  
and  
Governance



Security

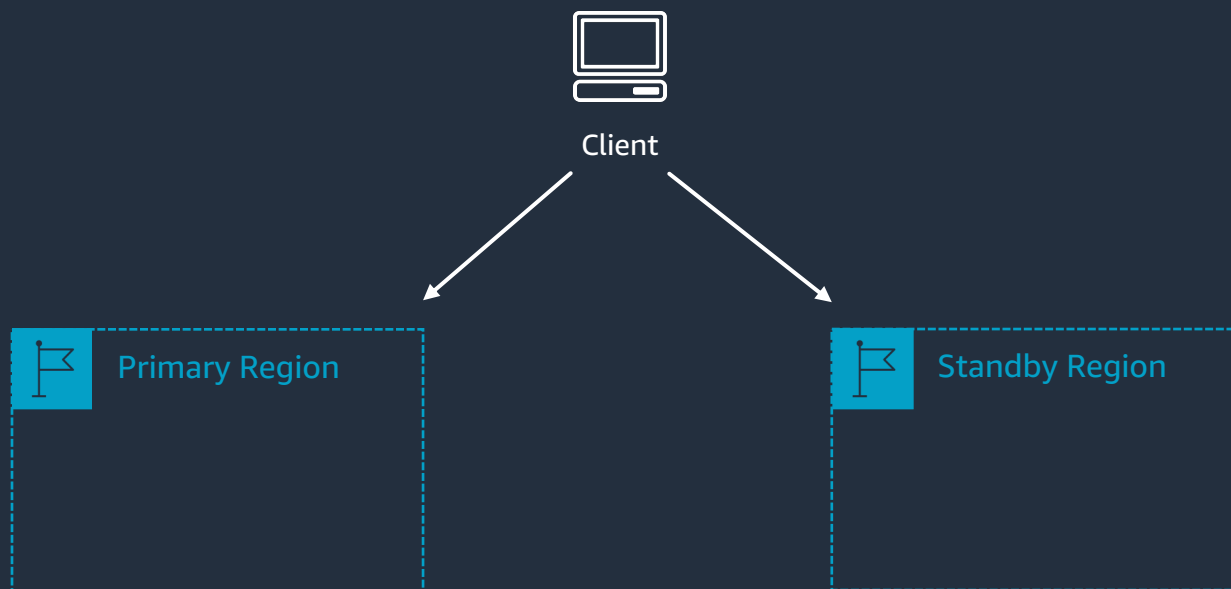
# オンプレミスおよびサードパーティへの依存関係

- ・ オンプレミスへの依存関係を特定する
- ・ ワークロード内のサードパーティ ソリューションを特定する



# フェイルオーバーメカニズム

- フェイルオーバーのためにプライマリリージョンに依存しない
- フェイルオーバーメカニズムのすべての依存関係を精査する

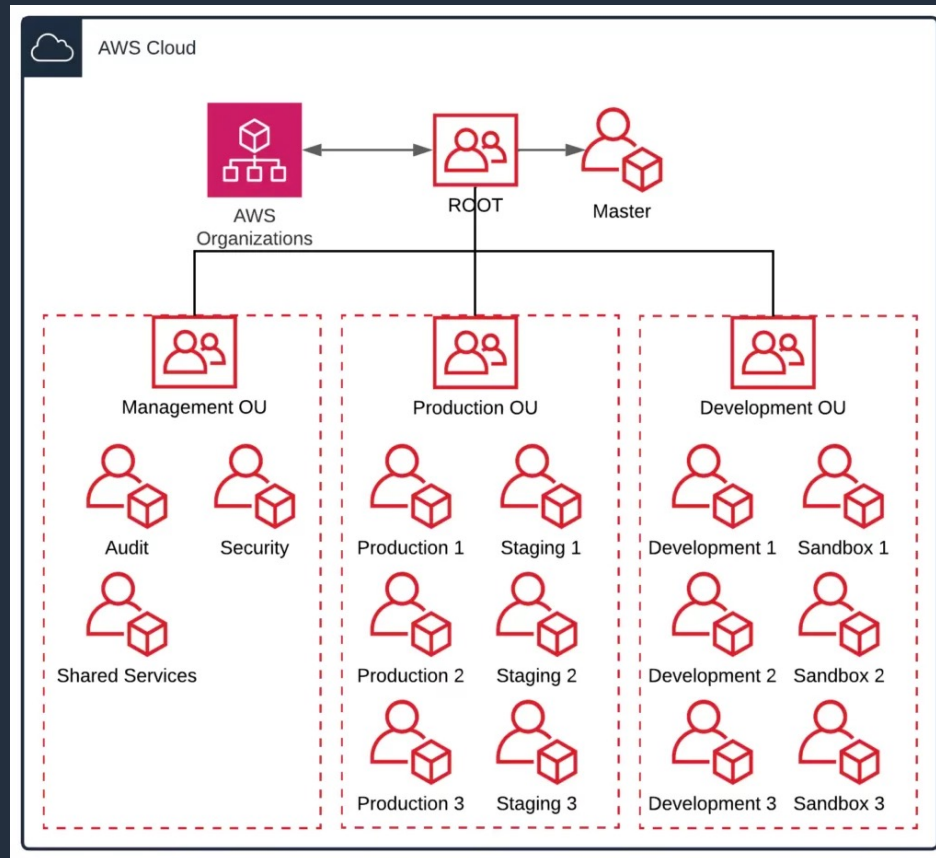


# マルチリージョン 基本

- 要件を理解する
- データを理解する
- 依存性を理解する
- オペレーションレディネス

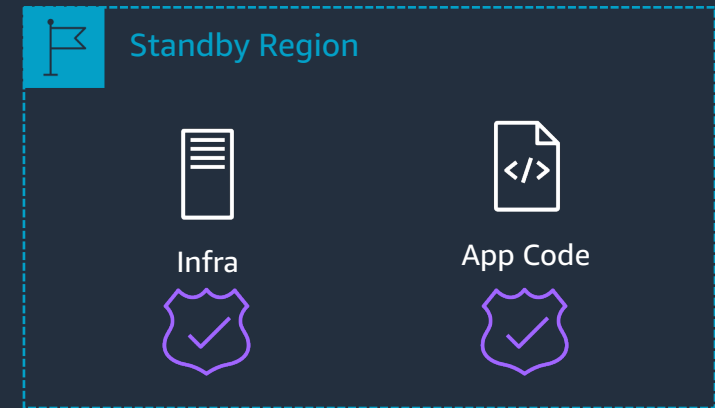
# AWS アカウントマネジメント

- AWS サービスクォータ
- IAM パーミッション
- サービスコントロールポリシー



# リージョンごとにデプロイする

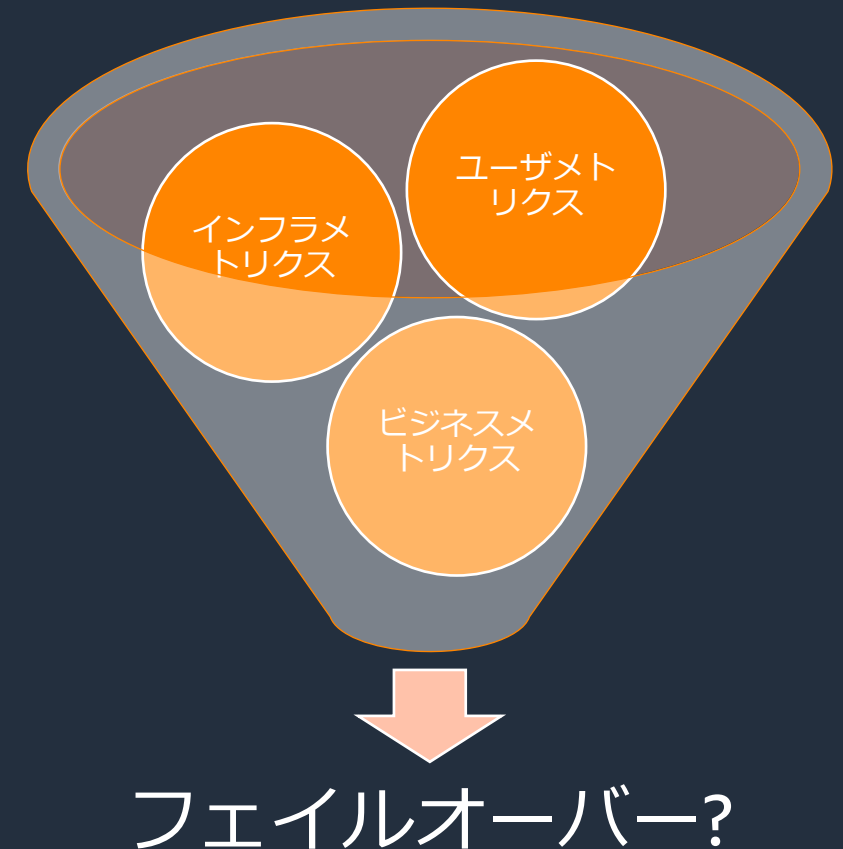
ブルーグリーンデプロイメント または カナリアデプロイメント





# プロセスと人

- テストされていない DR 戦略 = DR 戦略が無いのと等しい
- フェイルオーバーの範囲
  - 個々のマイクロサービス?
  - ビジネスの継続性?
- フェイルオーバー メカニズムを定期的にテストする必要があるだけでなく、
  - いつフェールオーバーするかのプロセスを定義する必要がある
- 運用継続計画の一部



# 参考：AWS Multi-Region Fundamentals

**AWS Multi-Region Fundamentals**  
AWS Whitepaper

**Abstract and introduction**  
Engineering and operating for resilience in a single Region  
Multi-Region fundamental 1: Understanding the requirements  
Multi-Region fundamental 2: Understanding the data  
Multi-Region fundamental 3: Understanding your workload dependencies  
Multi-Region fundamental 4: Operational readiness  
Conclusion  
Contributors  
Further reading  
Document revisions  
Notices  
AWS glossary

## AWS Multi-Region Fundamentals

[PDF](#) | [RSS](#)

Publication date: **December 20, 2022** ([Document revisions](#))

### Abstract

This advanced, 300-level paper is intended for cloud architects and senior leaders building workloads on AWS who are interested in using a multi-Region architecture to improve resilience for their workloads. This paper assumes baseline knowledge of AWS infrastructure and services. It outlines common multi-Region use cases, shares fundamental multi-Region concepts and implications around design, development, and deployment, and provides prescriptive guidance to help you better determine whether a multi-Region architecture is right for your workloads.

### Are you Well-Architected?

The [AWS Well-Architected Framework](#) helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the [AWS Well-Architected Tool](#), available at no charge in the [AWS Management Console](#), you can review your workloads against these best practices by answering a set of questions for each pillar.

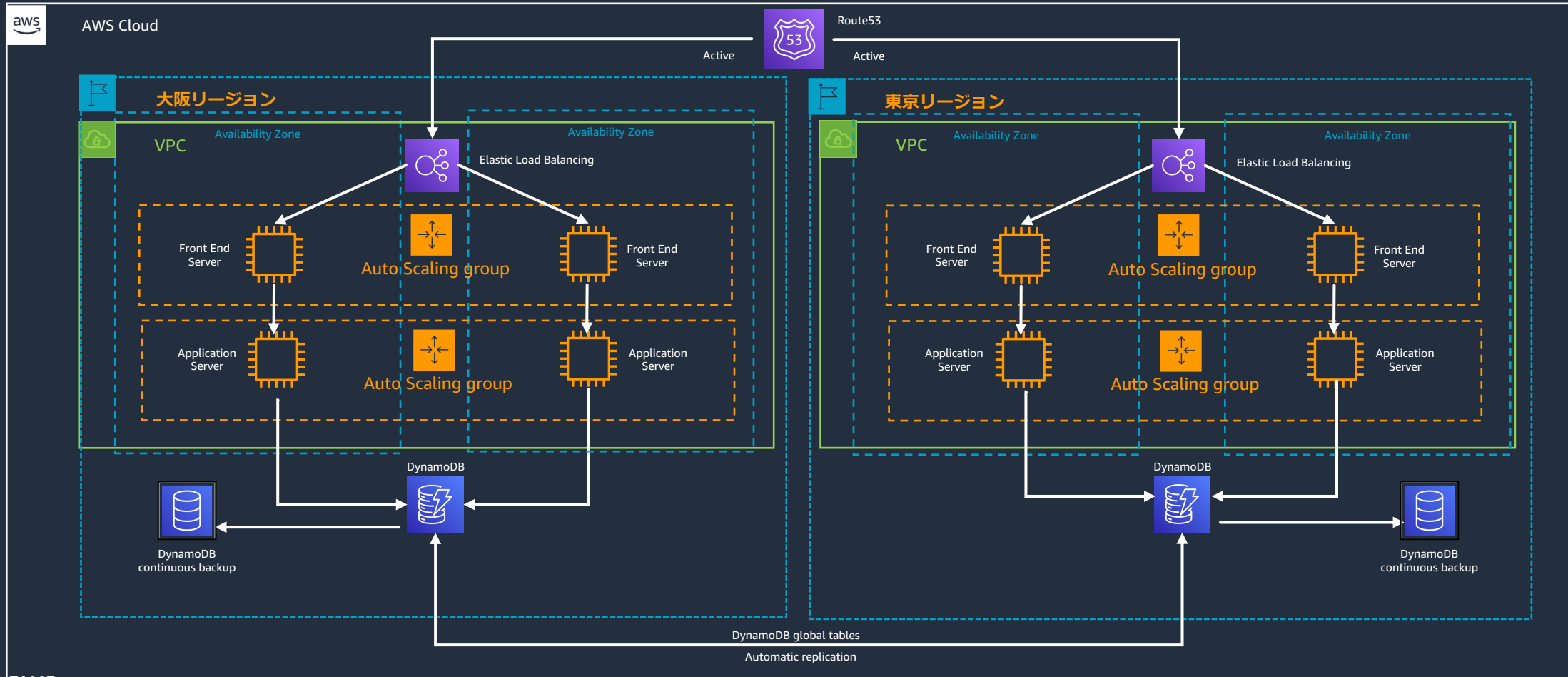
For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the [AWS Architecture Center](#).

### Introduction

<https://docs.aws.amazon.com/whitepapers/latest/aws-multi-region-fundamentals/aws-multi-region-fundamentals.html>

# マルチリージョンアーキテクチャの例

- 平常時からメイン サイト同等構成のサブサイトを運用
- 障害時は 自動フェイルオーバー



# 参考：マルチリージョンアーキテクチャパターン

ARC306-R1

## Multi-Region design patterns and best practices

Neeraj Kumar  
Principal Solutions Architect  
AWS

John Formento, Jr.  
Principal Solutions Architect  
AWS

Barry Sheward  
Chief Cloud Architect  
The Vanguard Group

AWS-50

## マルチリージョンでディザスタリカバリ(DR)戦略を検討するためのポイント

大場 崇令  
パートナー・アライアンス統括本部 ストラテジック SI 技術部  
シニアパートナーソリューションアーキテクト  
アマゾン ウェブ サービス ジャパン合同会社

aws © 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### Pattern #1: Active/passive

PILOT LIGHT, WARM STANDBY, ACTIVE/HOT STANDBY

- Use cases
  - Regional failover/fallback for DR/operational continuity
  - Regulatory requirements
- Key design considerations
  - Observability (health checks)
  - CI/CD, code and configuration drifts
  - Routing strategy
  - Failover/fallback procedures

### ウォームスタンバイ

- 平常時からメイン サイト同等機能をもち規模を縮小した サブサイトを運用
- 障害時は自動フェイルオーバーし、流量を絞って業務を継続

<https://www.youtube.com/watch?v=ilgpzLE7Hds>

<https://www.youtube.com/watch?v=bcVK3zN5Clc>

# アジェンダ

- なぜ Resilience が重要なのか？
- 責任共有モデル for Resilience
- Resilience のあるシステムを作るためのベストプラクティス
- マルチリージョン基本
- まとめ

# まとめ

- 責任共有モデル for Resilience
  - お客様の責任範囲：Resilience in the Cloud
  - AWS の責任範囲：Resilience of the Cloud
- Resilience を備えたシステムの設計ポイント
  - Well-Architected フレームワーク
  - マルチリージョンの基本
    - 要件を理解する
    - データを理解する
    - 依存性を理解する
    - オペレーションレディネス

# 本セッションのゴール

## ゴール

- Resilience（回復力）の責任共有モデルを理解できるようになる
- AWS上でResilience（回復力）を備えたシステムの設計ポイントが理解できるようになる



**Thank you!**